【Spring学习36】Spring事务(6)：声明式事务(集合Mybatis)

作者：soonfly (/authorarticle.html?author=soonfly) 2017-04-20 ☆ 收录到我的专题 (/select.html?articleId=401736)

标签 schema (http://www.bijishequ.com/info/search.html?searchText=schema) wwwspringframeworkorg (http://www.bijishequ.com/info/search.html?searchText=wwwspringframeworkorg) Spring (http://www.bijishequ.com/info/search.html?searchText=Spring) http (http://www.bijishequ.com/info/search.html?searchText=http) 事务 (http://www.bijishequ.com/info/search.html?searchText=事务)

Spring声明 式事务实现过程中，在配置文件中关于事务配置总是由三个组成部分，分别是DataSource、TransactionManager和代理机制这三部分，无论哪种配置方式，一般变化的只是代理机制这部分。DataSource、TransactionManager这两部分只是会根据数据访问方式有所变化。
比如使用Hibernate进行数据访问时，DataSource实际为SessionFactory，TransactionManager的实现为HibernateTransactionManager。使用myBatis时用的是JDBC事务管理器，因此TransactionManager的实现为DataSourceTransactionManager。

先假设我们在myBatis中自己写事务，是这个样子：

```
1  public class StudentService {
2  public Student createStudent(Student student) {
3  SqlSession sqlSession = MyBatisUtil.getSqlSessionFactory()
4  .openSession();
5  try {
6  StudentMapper mapper = sqlSession.getMapper(StudentMapper.class);
7  mapper.insertAddress(student.getAddress());
8  mapper.insertStudent(student);
9  sqlSession.commit();
10 return student;
11 } catch (Exception e) {
12 sqlSession.rollback();
13 throw new RuntimeException(e);
14 } finally {
15 sqlSession.close();
16 }
17 }
18 }
```

按这个方式，在每一个需要用到事务的方法中，添加事务的提交、回滚、关闭等。
现在我们来使用spring的事务处理能力。首先在Spring的配置文件中配置TransactionManager。
Spring全局配置beans.xml：

```
<?xml version="1.0" encoding="UTF-8" ?>
<beans xmlns="http://www.springframework.org/schema/beans" xmlns:context="http://www.springframework.org/schema/conte
xt" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:c="http://www.springframework.org/schema/c" xmlns:tx=
"http://www.springframework.org/schema/tx" xmlns:aop="http://www.springframework.org/schema/aop" xsi:schemaLocation=
"http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd http://www.
springframework.org/schema/context http://www.springframework.org/schema/context/spring-context-4.2.xsd http://www.sp
ringframework.org/schema/tx http://www.springframework.org/schema/tx/spring-tx-4.2.xsd http://www.springframework.or
g/schema/aop http://www.springframework.org/schema/aop/spring-aop-4.2.xsd">

<bean id="dataSource" class="org.springframework.jdbc.datasource.DriverManagerDataSource">
<property name="driverClassName" value="${jdbc.driverClassName}"></property>
<property name="url" value="${jdbc.url}"></property>
<property name="username" value="${jdbc.username}"></property>
<property name="password" value="${jdbc.password}"></property>
</bean>

<bean id="sqlSessionFactory" class="org.mybatis.spring.SqlSessionFactoryBean">
<property name="dataSource" ref="dataSource" />
<property name="typeAliases" value="com.owen.mybatis.domain.Student" />
<property name="typeHandlers" value="com.owen.mybatis.typehandlers.PhoneTypeHandler" />
<property name="typeHandlersPackage" value="com.owen.mybatis.typehandlers" />
<property name="mapperLocations" value="classpath*:com/mybatis3/**/*.xml" />
</bean>

<bean id="sqlSession" class="org.mybatis.spring.SqlSessionTemplate">
<constructor-arg index="0" ref="sqlSessionFactory" />
</bean>

<bean class="org.mybatis.spring.mapper.MapperScannerConfigurer">
<property name="basePackage" value="com.owen.mybatis.mappers" />
</bean>

<!-- ============事务配置============= -->
<!-- 基于注解的事务处理特性，Spring需要先使用下面的配置 -->
<tx:annotation-driven transaction-manager="transactionManager" />

<!-- 事务管理器 -->
<bean id="transactionManager" class="org.springframework.jdbc.datasource.DataSourceTransactionManager">
<property name="dataSource" ref="dataSource" />
</bean>
</beans>
```

现在可以在Spring的服务的Bean中注解@ Transactional。这个注解表明每个方法都是Spring来管理的。如果方法成功处理，那么Spring就会提交事务；如果就去处理过程出现了错误，那么事务就会被回滚。当然，Spring将会关心MyBatis的转换过程是否出现Exceptons的DataAccessExceptions的异常栈。
在DAO上需加上@Transactional注解，如下：

```
package twm.spring.transactiondemo.dao;

import java.util.List;
import org.hibernate.SessionFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.orm.hibernate3.support.HibernateDaoSupport;
import org.springframework.stereotype.Component;
import twm.spring.transactiondemo.pojo.User;

@Transactional(propagation = Propagation.REQUIRES_NEW, isolation = Isolation.READ_COMMITTED, noRollbackFor = { TExcep
tion.class }, readOnly = true, timeout = 3)
@Component
public class StudentService {
@Autowired
private StudentMapper studentMapper;

public Student createStudent(Student student) {
studentMapper.insertAddress(student.getAddress());
if(student.getName().equalsIgnoreCase("")) {
throw new RuntimeException("Student name should not be empty.");
}
studentMapper.insertStudent(student);
return student;
}
}
```